به نام خدا

دانشگاه تهران

دانشکده مهندسی برق و کامپیوتر

پروژه کامپیوتری

جبر خطی

عرفان باقری سولا

شماره دانشجویی: ۸۱۰۱۹۸۳۶۱

جدول مطالب

۱. پیاده سازی چند الگوریتم

۱-۱ درونیابی لاگرانژ

با استفاده از درونیابی لاگرانژ، منحنی های تخمین زده شده را به ازای چند جمله ای های درجه ۳ تا ۶ به دست می آوریم:

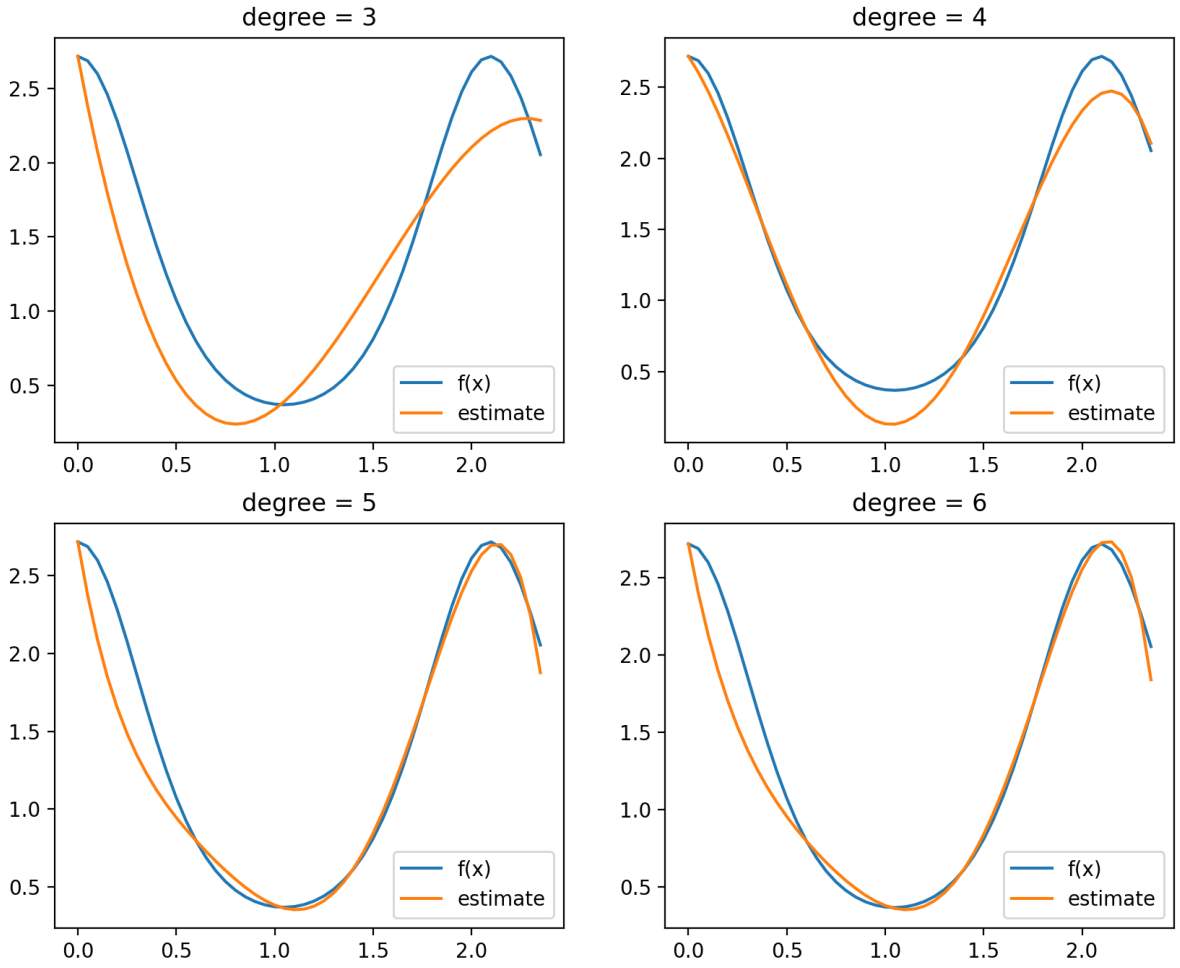
$$-1.276\,x^3 + 5.903\,x^2 - 7.007\,x + 2.718$$

$$-1.447\,x^4 + 5.821\,x^3 - 4.938\,x^2 - 2.023\,x + 2.718$$

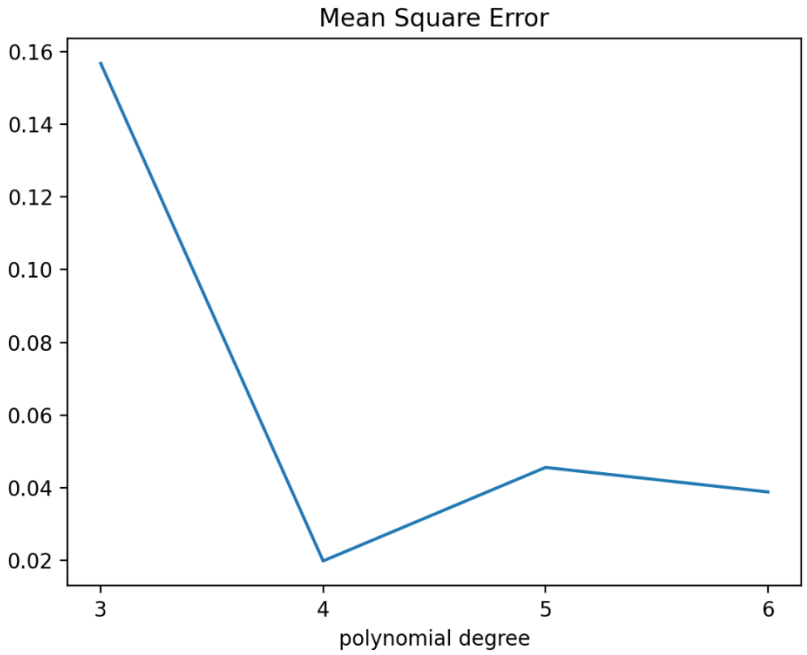
$$-1.638\,x^5 + 8.448\,x^4 - 15.352\,x^3 + 13.735\,x^2 - 7.529\,x + 2.718$$

$$-0.199\,x^6 - 0.230\,x^5 + 4.636\,x^4 - 10.431\,x^3 + 10.728\,x^2 - 6.840\,x + 2.718$$

در ادامه نمودار منحنی های تخمین زده شده را در کنار منحنی اصلی مشاهده می کنیم:



نمودار خطا به ازای درجه های مختلف به صورت زیر می باشد:



با توجه به نمودار بالا واضح است که چند جمله ای درجه ۴ بهترین منحنی برای تخمین منحنی اصلی می باشد.

۲-۱ تجزیه LU برای ماتریس بالا هسنبرگی

تابع مد نظر را پیاده سازی کرده و آن را روی یک ماتریس بالا هسنبرگی دلخواه تست می کنیم.

```
array([[ 4,  5,  6],   array([[ 1.,  0.,  0.],   array([[ 4.,  5.,  6.],
       [ 8,  7, 13],          [ 2.,  1.,  0.],          [ 0., -3.,  1.],
       [ 0, 12,  4]])         [ 0., -4.,  1.]])         [ 0.,  0.,  8.]])
```

$$M = LU$$

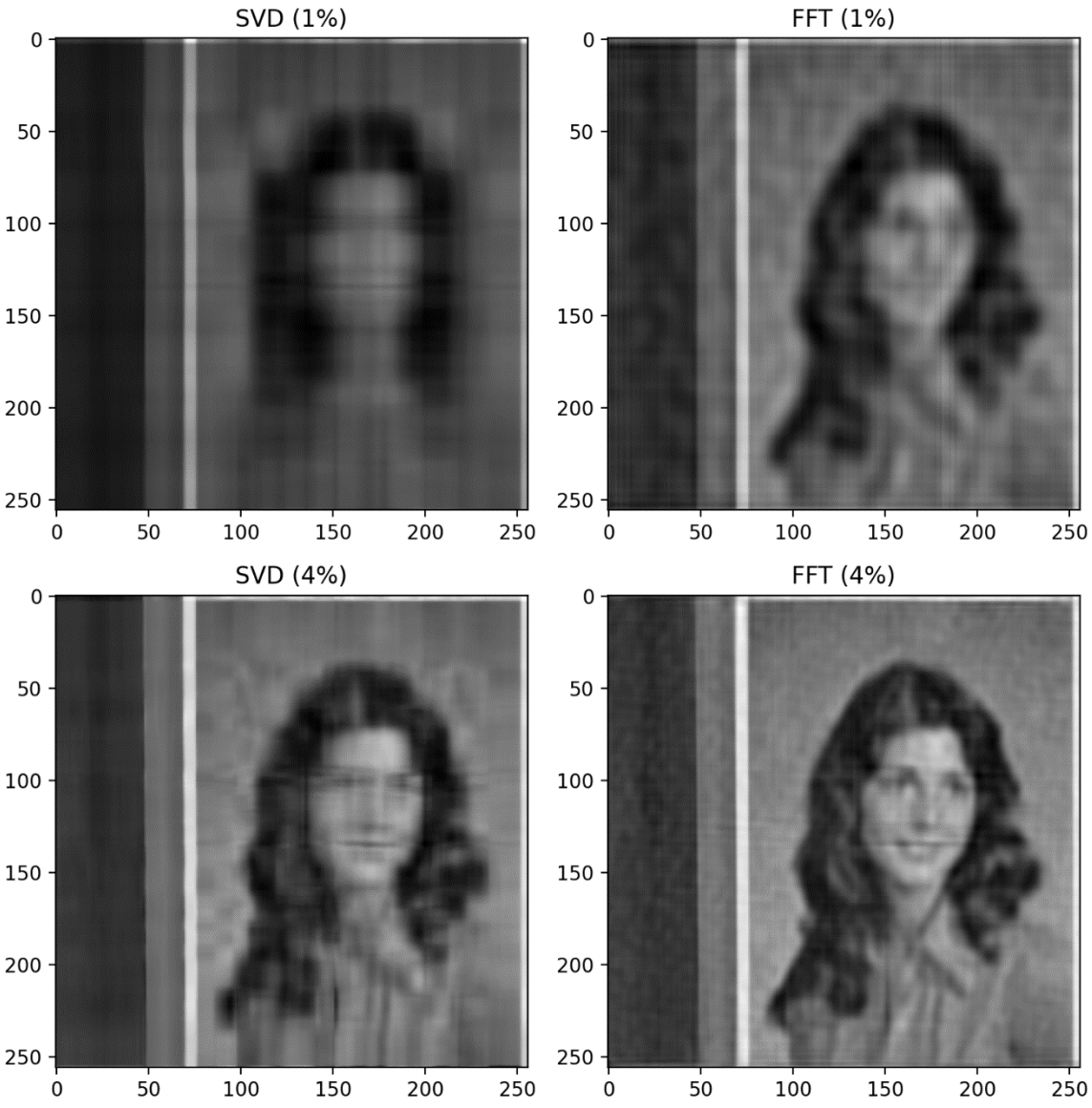
می بینیم که تجزیه LU به درستی انجام شده است.

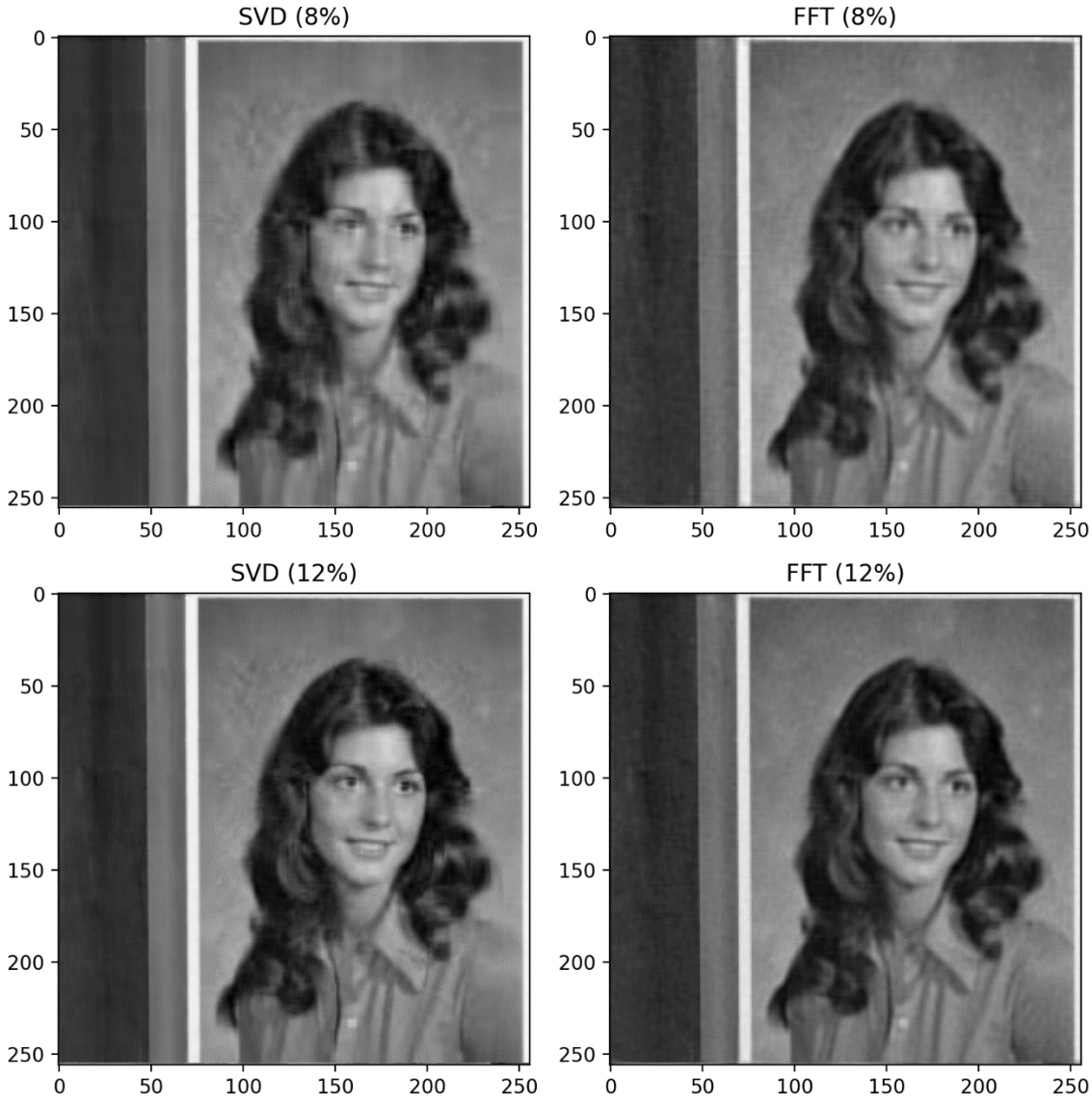
پیچیدگی محاسباتی تجزیه LU عادی از مرتبه n^3 می باشد.
این یعنی برای یک ماتریس مربعی، تعداد floating point operation مورد نیاز برای تجزیه آن، با توان ۳ نسبت به ابعاد آن رشد خواهد کرد. حال اگر ماتریس بالا هسنبرگی باشد، با استفاده از الگوریتمی که پیاده سازی کردیم می توان پیچیدگی محاسباتی را به مرتبه n^2 کاهش داد.

واضح است که هزینه تابع نوشته شده هم به تعداد سطر و هم به تعداد ستون وابسته است چرا که باید با استفاده از درایه روی قطر اصلی هر سطر، درایه روی قطر فرعی سطر پایینی را صفر کرد (وابستگی به تعداد سطر). برای این کار ضریبی از سطر بالایی با سطر پایینی جمع می شود و واضح است تعداد عملیات مورد نیاز برای جمع دو سطر وابسته به تعداد درایه های هر سطر یعنی تعداد ستون ها نیز می باشد.

۲. فشرده سازی SVD و FFT

با توجه به این که ابعاد تصویر 256x256 می باشد، ماتریس آن دارای ۲۵۶ مقدار تکین می باشد و ضریب فشردگی ۰/۱ درصد معادل انتخاب ۰/۲۵۶ تا از مقادیر تکین در فشرده سازی SVD می باشد که عملا ناممکن است. به همین دلیل فشرده سازی را برای ضرایب [1, 4, 8, 12] انجام می دهم. (به هر حال از کلیت مسئله کاسته نمی شود)





ملاحظه می کنیم که برای ضرایب فشرده سازی کوچکتر، عملکرد FFT به مراتب بهتر از SVD می باشد. با افزایش ضریب فشرده سازی، نتایج SVD سریعتر از FFT بهبود می یابد به طوری که در ضریب فشرده سازی ۱۲ درصد، نتیجه SVD عملکردی مشابه و یا حتی کمی بهتر از FFT دارد.

به طور کلی چون FFT از حوزه فرکانسی تصاویر برای فشرده سازی آنها استفاده می کند، حتی در ضرایب فشرده سازی خیلی پایین نیز می تواند یک شمای کلی قابل قبول از تصویر را ارائه دهد در صورتی که علکرد SVD در ضرایب فشرده سازی خیلی پایین مانند ۱ درصد به مراتب بدتر است. البته FFT نیز به خاطر داشتن مقادیر مختلط، حافظه بیشتری نیاز دارد.

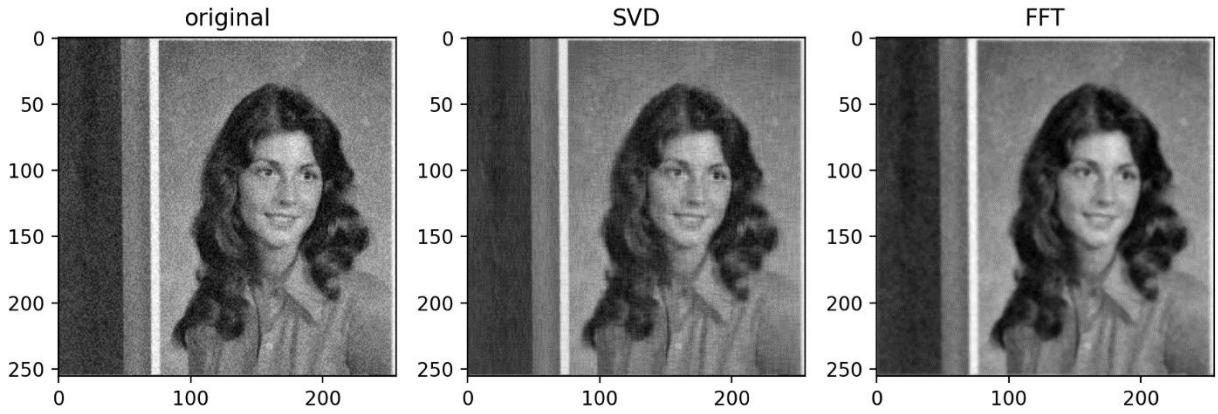
روش PCA یک الگوریتم برای کاهش بعد داده ها است به طوری که فضای با بعد پایین تر جدید، بیشترین اطلاعات را از فضای با بعد بالاتر در خود نگه دارد. روش PCA شبیه روش SVD است با این تفاوت که در فشرده سازی SVD فقط با در نظر گرفتن یک داده عملیات کاهش بعد انجام می شود ولی در روش PCA با در نظر گرفتن یک مجموعه از داده ها عملیات کاهش بعد انجام می شود تا واریانس نمونه ها ماکسیمم شود.

برای این که الگوریتم های بالا را روی تصاویر رنگی اعمال کنیم، کافی است روی هر کانال به طور جداگانه فشرده سازی را انجام دهیم. هر دو به ازای ضرایب فشرده سازی مناسب خود که پیشتر بحث کردیم، نتیاج قابل قبولی می دهند.

برای denoise کردن تصاویر، الگوریتم مبتنی بر SVD همانند الگوریتم فشرده سازی خواهد بود چرا که انتظار داریم با انتخاب بردار های پایه ای که مقدار تکین مربوط به آنها بیشتر است، اطلاعات کلی تر تصویر را نگه داریم و اطلاعات کم اهمیت تر مانند نویز را نداشته باشیم.

الگوریتم مبتنی بر FFT نیز با توجه به این که اطلاعات و ساختار کلی تصاویر در فرکانس های پایین و نویز در فرکانس های بالا ایجاد می شود، می تواند با یک فیلتر پایین گذر نویز های تصویر را حذف کند. هرچه پهنای باند فیلتر پایین گذر کمتر باشد، تصویر تار تر می شود. برای پیاده سازی فیلتر پایین گذر از یک کرنل گاوسی دو بعدی استفاده می کنیم که آن را در ضرایب حوزه فرکانسی تصویر ضرب می کنیم. در نتیجه ضرایبی که در مختصات دور تری از مرکز باشند (مربوط به فرکانس بالاتری باشند)، بسیار تضعیف می شوند.

نتایج حاصل از denoise را برای هر دو روش در ادامه مشاهده می کنیم.



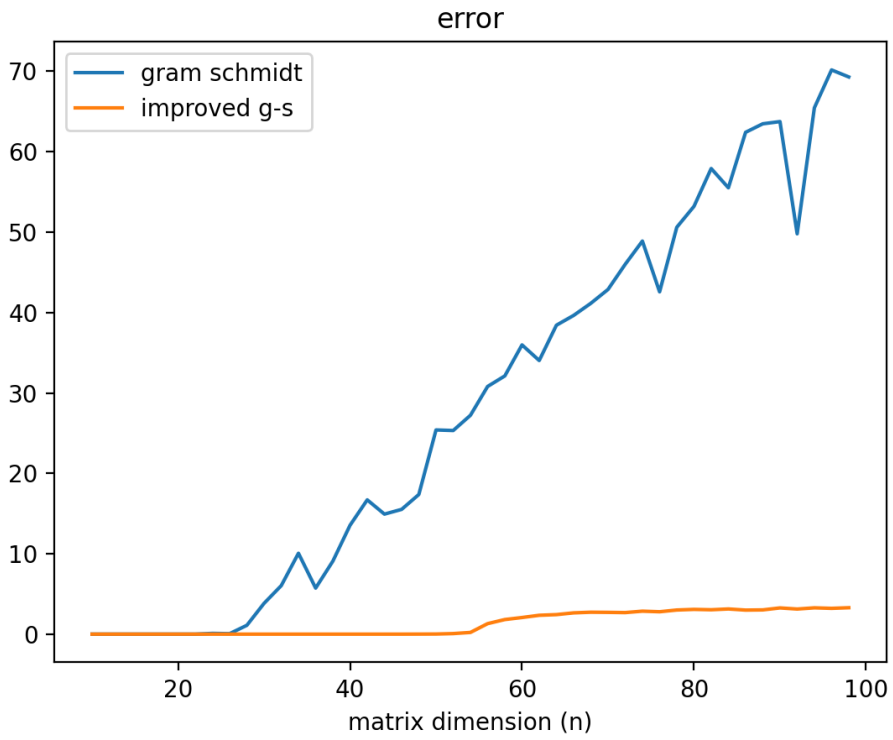
به طور کلی FFT با توجه به انعطاف زیادی که در انتخاب کرنل در اختیارمان می گذارد، عملکرد بهتری در denoise کردن تصاویر دارد.

۳. تطبیق هیستوگرام

تابعی پیاده سازی می کنیم که با گرفتن تصاویر یک کاناله مبدا و مرجع، تصویر مقصد را با استفاده از روش تطبیق هیستوگرام محاسبه کرده و برگرداند. حال با استفاده از این تابع و برای هر سه کانال RGB تطبیق هیستوگرام را روی ۲ تصویر داده شده انجام می دهیم. خروجی به صورت زیر خواهد بود:

۴. گرام اشمیت تغییر یافته

توبع مربوط به گرام اشمیت و گرام اشمیت تغییر یافته را پیاده سازی می کنیم سپس برای ورودی های مختلف عملکرد آنها را بررسی می کنیم. نمودار خطای به دست آمده به ازای ابعاد ماتریس ورودی به این صورت است:



ملاحظه می کنیم که خطای الگوریتم گرام اشمیت تقریبا به صورت خطی با افزایش ابعاد ماتریس ورودی افزایش پیدا می کند در حالی که الگوریتم گرام اشمیت تغییر یافته در مقابل خطا پایداری بسیار بیشتری دارد و خطای آن به کندی افزایش پیدا می کند. به عبارتی خطای الگوریتم تغییر یافته مرتبه ای کمتر از مرتبه خطی دارد که این عملکرد بسیار مطلوب است.